An iteration method for solving the linear system $Ax = b$

Zhaolu Tian[a,*], Maoyi Tian[b], Yan Zhang[b,*], Pihua Wen[c]

[a] *College of Data Science, Taiyuan University of Technology, Taiyuan 030024, P.R.China*

[b] *Geomatics College, Shandong University of Science and Technology, Qingdao 266590, P.R.China*

[c] *School of Engineering and Materials Science, Queen Mary University of London, London E14NS, UK*

Abstract

In this paper, based on a convergence splitting of the matrix A, we present an inner-outer iteration method for solving the linear system $Ax = b$. We analyze the overall convergence of this method without any other restriction on its parameters. Moreover, we give the accelerated inner-outer iteration method, and discuss how to apply the inner-outer iterations as a preconditioner for the krylov subspace methods. The inner-outer iteration method can also be used for the solution of $AXB = C$. Finally, several numerical examples are given to validate the performance of our proposed algorithms.

Key words: *Inner-outer iterations; Convergence; Accelerate; Preconditioned*

1. Introduction

Consider the solution of the following linear system

$$Ax = b, \tag{1.1}$$

where $A \in R^{n\times n}$ is a nonsingular matrix, and $x, b \in R^n$.

The linear system (1.1) plays an important role in scientific and engineering computations. Many iteration methods have been proposed for solving (1.1) for decades, for example, the Jacobi, Gauss-Seidel and the Successive Overrelaxation (SOR) methods [1,6,10], etc. Krylov subspace methods, such as GMRES [7] and CG [5] have been considered as classical iteration methods for large and sparse linear systems. However, Krylov subspace methods may converge very slowly or not at all for many cases arising in certain computational mechanics and electronic device simulation problems. Hence preconditioning techniques [19,20,22,23] have been widely used to improve the convergent behavior of Krylov subspace methods. For large non-Hermitian positive definite linear system, the HSS method [2,4,8,11,15] has attracted more attention due to its promising performance and elegant mathematical properties.

In [3] the authors proposed an effective iteration method for PageRank computation [17,18,21] by using the inner-outer stationary iterations, which is very simple, and can be implemented and parallelized in a straightforward fashion, furthermore, this algorithm is an excellent combination of minimal memory requirements and fast convergence. In this paper, we will consider the

*Corresponding author.

Tel/fax:+86 0532 80698369.

E-mail address: tianzhaolu2004@126.com (Z.Tian), tmyzhangyan@126.com(Y.Zhang).

similar iteration method for the linear system (1.1). Moreover, the same idea can be used as a preconditioner for a nonstationary iteration method such as GMRES method.

The remainder of this paper is organized as follows: In section 2 we give the inner-outer iteration method and corresponding algorithm, discuss its convergence in detail. In section 3 we consider the overall convergence of the inner-outer iteration method and give a comparison result. In section 4 we present the accelerated inner-outer iteration method, and analyze how to use the inner-outer iteration as a preconditioning technique for nonstationary iteration methods. In section 5 we apply the inner-outer iteration method for solving the matrix equation $AXB = C$ and propose the corresponding algorithm. Several numerical examples are given in section 6 to illustrate the effectiveness of our proposed algorithms. Finally, we draw some conclusions in section 7.

2. The inner-outer iteration method for solving (1.1)

2.1 The inner-outer iteration method

Let $A = M - N$ be a convergent splitting, then we obtain the following linear system equivalent to (1.1):

$$(I - R)x = c, \tag{2.1}$$

where $R = M^{-1}N$ and $c = M^{-1}b$.

The inner-outer iteration method for solving (2.1) is expressed as follows:

$$(I - \alpha R)x_{k+1} = (1 - \alpha)Rx_k + c, \ k = 0, 1, 2, \cdots \tag{2.2}$$

with $0 < \alpha < 1$. Here we regard (2.2) as the outer iteration. It is difficult to obtain x_{k+1} directly from (2.2) since it is equivalent to solving (1.1), then we approximate x_{k+1} by the following inner iteration. Let $g = (1 - \alpha)Rx_k + c$, and we define the inner linear system as

$$(I - \alpha R)z = g, \tag{2.3}$$

then apply the inner iteration

$$z_{s+1} = \alpha Rz_s + g, \ s = 0, 1, 2, \cdots, l - 1, \tag{2.4}$$

where we take $z_0 = x_k$ as the initial value and assign z_l as the new x_{k+1}. If we have not converged to the desired solution of (2.1), then we repeat the above procedure, increasing k and using x_k as the new initial value z_0.

Assume the parameters η and ε are the tolerances of the inner and outer iterations, respectively, and we use the 2-norm of the residuals of the outer system (2.1) and the inner system (2.3) as stopping criterions to terminate the iterations. For the outer system (2.1) we have

$$\|c - (I - R)x_{k+1}\|_2 < \varepsilon,$$

and for the inner system (2.3) we apply

$$\|g - (I - \alpha R)z_{s+1}\|_2 < \eta.$$

Algorithm 1: The inner-outer iteration method
Input: R, c, α, ε, η

Output: x
1: $x \leftarrow c$
2: $z \leftarrow Rx$
3: **while** $\|c - x + z\|_2 \geq \varepsilon$
4: $\quad g \leftarrow (1-\alpha)z + c$
5: $\quad$ **repeat**
6: $\quad\quad x \leftarrow g + \alpha z$
7: $\quad\quad z \leftarrow Rx$
8: $\quad$ **until**$\|g + \alpha z - x\| < \eta$
9: **end while**
10: $x \leftarrow z + c$

In this algorithm lines 1 and 2 initialize $x = c$ and $z = Rx$. g in (2.3) is obtained in line 4. The inner iteration defined in (2.4) is implemented from line 5 to line 8. To terminate the Algorithm 1, line 3 tests the residual of the outer iteration (2.2), and for the inner iteration, the stopping criteria is in line 8. The vector $(I - \alpha R)z_{j+1}$ is given by $x - \alpha z$, since now x holds z_{j+1} and z is computed by Rx in line 7 as well as in line 2. Upon exit from the algorithm, x is the desired approximation to the exact solution of the Eq.(2.1). In line 10 we execute a single iteration, since z is already updated and we can use it as well.

Let $A = D - L - U$, where D is the diagonal of A, $-L$ is the strictly lower triangular part of A, and $-U$ is the strictly upper triangular part of A, respectively. Then the matrix splitting of the AOR iteration method [24] for solving (1.1) is

$$M_A = \frac{1}{\omega}(D - \gamma L),\ N_A = \frac{1}{\omega}((1-\omega)D + (\omega - \gamma)L + \omega U), \tag{2.5}$$

where $\omega,\ \gamma$ are two real parameters with $\omega \neq 0$.

For different ω and γ, we can obtain the corresponding iteration methods:

(1) the Jacobi method: $\omega = 1,\ \gamma = 0$.

(2) the Gauss-Seidel method: $\omega = 1,\ \gamma = 1$.

(3) the SOR method: $\omega = \gamma$.

Then we can obtain the inner-outer iteration method based on the AOR splitting (2.5), and for different $\omega,\ \gamma$ we can get the corresponding inner-outer iteration methods.

In Example 1 the coefficient matrix A of (1.1) is a nonsingular $M-$ matrix, then we need to discuss the convergence interval of the AOR iteration for this case, which will contribute to the choices of the parameters ω and γ. From [25] we can obtain the convergence theorem of the AOR iteration for a nonsingular $M-$ matrix A.

Theorem 2.1. Let $G_A = M_A^{-1}N_A$ and $J = D^{-1}(L+U)$ be the AOR iteration matrix and Jacobi iteration matrix for (1.1), respectively. If A is a nonsingular $M-$ matrix, then

$$\rho(G_A) \leq |1 - \omega| + \omega\rho(J) < 1$$

for $0 < \omega < \frac{2}{1+\rho(J)}$ and $0 \leq \gamma \leq \omega$.

Proof. This is a special case of Theorem 3.3 [25]. □

In order to improve the convergence behaviors of the iteration methods, many preconditioners [19,20,22,23] have been proposed for solving the linear system (1.1).

Firstly transforming (1.1) into the following preconditioned form

$$PAx = Pb, \tag{2.6}$$

where $P \in R^{n\times n}$ is a preconditioner and nonsingular.

Let $PA = M_p - N_p$ be a convergence splitting and M_p is nonsingular, then we have from (2.6)

$$(I - R_p)x = \bar{c}, \tag{2.7}$$

where $R_p = M_p^{-1}N_p,\ \bar{c} = M_p^{-1}Pb$. We can use the inner-outer iteration method (Algorithm 1) to solve (2.7), which is called the preconditioned inner-outer iteration method.

Remark 1. For the inner-outer iteration method we do not consider the case $\alpha = 0$ or $\alpha = 1$. If $\alpha = 0$, the outer iteration (2.2) would simply lead back to the iteration sequence $x_{k+1} = Rx_k + c$, and the inner iteration (2.4) is meaningless. For the case $\alpha = 1$, the inner iteration (2.4) is also equivalent to the iteration sequence $x_{k+1} = Rx_k + c$.

2.2 Convergence analysis of the inner-outer iteration method

In this section, we will discuss the convergence of the inner-outer iteration method. Here we will analyze the convergence of the iterations (2.2) and (2.4), respectively.

Theorem 2.2 [6]. The iteration sequence $x_{m+1} = Rx_m + c$ converges to the solution of $Ax = b$ for all starting vectors x_0 and for all b if and only if $\rho(R) < 1$.

The outer iteration (2.2) is associated with the matrix splitting

$$I - R = M_1 - N_1,\ M_1 = I - \alpha R,\ N_1 = (1-\alpha)R,$$

and the corresponding outer iteration matrix is

$$R_1 = M_1^{-1}N_1 = (1-\alpha)(I - \alpha R)^{-1}R. \tag{2.8}$$

The inner iteration for solving (2.3) has the following matrix splitting

$$M_1 = I - \alpha R = M_2 - N_2,\ M_2 = I,\ N_2 = \alpha R,$$

and the inner iteration matrix is

$$R_2 = \alpha R. \tag{2.9}$$

Theorem 2.3. Given $\rho(R) < 1$ and $0 < \alpha < 1$, then the outer iteration (2.2) is convergent, where $\rho(A)$ denotes the spectral radius of matrix A.

Proof. Let λ_i be an eigenvalue of R, then $|\lambda_i| \le \rho(R) < 1$ from the assumption. Assume that μ_i is an eigenvalue of R_1 in (2.8), then

$$\mu_i = \frac{(1-\alpha)\lambda_i}{1 - \alpha\lambda_i}. \tag{2.10}$$

Since $|\lambda_i| \le \rho(R) < 1$ and $0 < \alpha < 1$, then from (2.10) we have

$$|\mu_i| = \left|\frac{(1-\alpha)\lambda_i}{1-\alpha\lambda_i}\right| \le \frac{(1-\alpha)|\rho(R)|}{1-\alpha|\rho(R)|} = \frac{1-\alpha}{|\rho(R)|^{-1} - \alpha} < 1. \tag{2.11}$$

From Theorem 2.2 and (2.11) the outer iteration (2.2) is convergent. □

Theorem 2.4. If $\rho(R) < 1$ and $0 < \alpha < 1$, then the inner iteration (2.4) is convergent.

Proof. Let ϕ_i be an eigenvalue of R_2 in (2.9), then

$$\phi_i = \alpha\lambda_i, \tag{2.12}$$

where λ_i is an eigenvalue of R. Then from (2.12) we get

$$|\phi_i| = |\alpha\lambda_i| \leq \alpha\rho(R) < 1. \tag{2.13}$$

From Theorem 2.2 and (2.13) we complete the proof. □

Remark 2. Let $A = \bar{M}_i - \bar{N}_i$ $(i = 1, 2)$ be two convergence splittings. If $\rho(\bar{M}_1^{-1}\bar{N}_1) < \rho(\bar{M}_2^{-1}\bar{N}_2) < 1$, then from Theorems 2.3 and 2.4 it is clear that the eigenvalues of the iteration matrices $\bar{R}_1$ and $\bar{R}_2$ based on $(\bar{M}_1, \bar{N}_1)$ have a smaller upper bound, then the inner-outer iteration method generated by $(\bar{M}_1, \bar{N}_1)$ maybe converges faster than that derived from $(\bar{M}_2, \bar{N}_2)$.

3. The overall convergence of the inner-outer iteration method

In this section, we will give the proof of the overall convergence for the inner-outer iteration method without considering the parameters ε and η , which shows that the inner-outer iteration method converges linearly to the exact solution x^* of the linear system (2.1).

Lemma 3.1 [6]. For all operator norms $\rho(R) \leq \|R\|$. For all R and for all $\varepsilon > 0$ there is an operator norm $\|R\|_\star \leq \rho(R) + \varepsilon$. The norm $\|\cdot\|_\star$ depends on both R and ε, where $\rho(C)$ denotes the spectral radius of the matrix C.

Lemma 3.2 [6]. Let $\|\cdot\|$ satisfy $\|AB\| \leq \|A\| \cdot \|B\|$. Then $\|X\| < 1$ implies that $I - X$ is invertible, $(I - X)^{-1} = \sum_{i=0}^{\infty} X^i$, and $\|(I - X)^{-1}\| \leq \frac{1}{1-\|X\|}$.

Firstly we rewrite the inner-outer iteration method as a two-stage iteration framework [17,32]:

$$\begin{cases} x_{k,0} = x_k, x_0 = c,\ x_{k+1} = x_{k,m_k}, \\ x_{k,j+1} = \alpha R x_{k,j} + (1-\alpha)R x_k + c,\ k = 0, 1, 2, \cdots, \\ j = 0, 1, 2, \cdots, m_k - 1. \end{cases} \tag{3.1}$$

Theorem 3.1. Let $A = M - N$ be a convergence splitting and $0 < \alpha < 1$, and m_k be the number of the inner iteration steps at the $k-$th outer iteration. Then the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by (3.1) converges to the exact solution x^* of (2.1), and faster than the iteration sequence derived from (2.1) for the same initial value x_0.

Proof. From (3.1) it follows that

$$x_{k,j+1} = \left[(\alpha R)^{j+1} + (1-\alpha)\sum_{s=0}^{j}(\alpha R)^s R\right] x_k + \sum_{s=0}^{j}(\alpha R)^s c.$$

Then we have

$$x_{k+1} = H_k x_k + E_k c, \quad k = 0, 1, 2, \cdots, \tag{3.2}$$

where $H_k = (\alpha R)^{m_k} + (1-\alpha)\sum_{s=0}^{m_k-1}(\alpha R)^s R$ and $E_k = \sum_{s=0}^{m_k-1}(\alpha R)^s$.

Since x^* is the exact solution of (1.1), then from (3.2) we obtain

$$x^* = H_k x^* + E_k c, \quad k = 0, 1, 2, \cdots. \tag{3.3}$$

Subtracting (3.3) from (3.2), then

$$x_{k+1} - x^* = H_k(x_k - x^*) = \cdots = H_k H_{k-1} \cdots H_0(x_0 - x^*), \quad k = 0, 1, 2, \cdots$$

and

$$\begin{aligned} H_k &= (\alpha R)^{m_k} + (1-\alpha)\textstyle\sum_{s=0}^{m_k-1}(\alpha R)^s R \\ &= (\alpha R)^{m_k} - \textstyle\sum_{s=0}^{m_k-1}(\alpha R)^s[(I-R)-(I-\alpha R)] \\ &= (\alpha R)^{m_k} + (I-(\alpha R)^{m_k}) - \textstyle\sum_{s=0}^{m_k-1}(\alpha R)^s(I-R) \\ &= I - \textstyle\sum_{s=0}^{m_k-1}(\alpha R)^s(I-R). \end{aligned} \tag{3.4}$$

Let λ_i be an eigenvalue of R, then from(3.4) we have

$$\phi_i^{(k)} = 1 - \frac{(1-\lambda_i)(1-(\alpha\lambda_i)^{m_k})}{1-\alpha\lambda_i} \tag{3.5}$$

is an eigenvalue of H_k.

Since $0 < \alpha < 1$ and $|\lambda_i| < 1$, then from (3.5) it is clear that

$$\begin{aligned} \left|\phi_i^{(k)}\right| &= \left|1 - \tfrac{(1-\lambda_i)(1-(\alpha\lambda_i)^{m_k})}{1-\alpha\lambda_i}\right| \\ &= \left|\tfrac{\lambda_i(1-\alpha+\alpha^{m_k}\lambda_i^{m_k-1}-(\alpha\lambda_i)^{m_k})}{1-\alpha\lambda_i}\right| \\ &< \tfrac{|1-\alpha|+|\alpha\lambda_i|^{m_k-1}|1-\lambda_i|\alpha}{|1-\alpha\lambda_i|} \\ &< \tfrac{|1-\alpha|+|\alpha\lambda_i|^{m_k-1}(1+\rho(R))\alpha}{|1-\alpha\lambda_i|} \\ &= \tfrac{|1-\alpha|}{|1-\alpha\lambda_i|} < 1 \end{aligned} \tag{3.6}$$

as $m_k \to \infty$. Then $\rho(H_k) < 1$.

Let $\psi = \max\limits_k\{\rho(H_k)\} < 1$ $(k = 0, 1, 2, \cdots)$ and ν_i be an eigenvalue of $H_k H_{k-1}\cdots H_0$, then we get

$$\nu_i = \Pi_{s=0}^k \phi_i^{(s)} = \Pi_{s=0}^k \left(1 - \frac{(1-\lambda_i)(1-(\alpha\lambda_i)^{m_s})}{1-\alpha\lambda_i}\right),$$

so

$$\rho(H_k H_{k-1}\cdots H_0) = \max_i\{|\nu_i|\} = \max_i\{\Pi_{s=0}^k|\phi_i^{(s)}|\} \le \rho(H_k)\rho(H_{k-1})\cdots\rho(H_0) \le \psi^{k+1} < 1.$$

From Lemma 3.1 there exists an operator norm $\|\cdot\|_\varrho$ such that

$$\|H_k H_{k-1}\cdots H_0\|_\varrho \le \psi^{k+1} + \varepsilon$$

for $\forall\varepsilon > 0$. Then we have

$$\|x_{k+1} - x^*\|_\varrho \le \|H_k H_{k-1}\cdots H_0\|_\varrho \|x_0 - x^*\|_\varrho \le \left(\psi^{k+1} + \varepsilon\right)\|x_0 - x^*\|_\varrho. \tag{3.7}$$

So from (3.7) we learn that the iteration sequence (3.1) converges to the exact solution x^* as $\varepsilon \to 0$. □

From (3.6) it is clear that

$$\begin{aligned} \left|(1-\alpha\lambda_i)\phi_i^{(k)}\right| &= \left|\lambda_i(1-\alpha+\alpha^{m_k}\lambda_i^{m_k-1}-(\alpha\lambda_i)^{m_k})\right| \\ &= |\lambda_i|\cdot|1-\alpha+\alpha(\alpha\lambda_i)^{m_k-1}(1-\lambda_i)| \\ &< |\lambda_i|(|1-\alpha|+2\alpha|\alpha\lambda_i|^{m_k-1}) \\ &= |\lambda_i|\cdot|1-\alpha| \\ &< |\lambda_i|\cdot|1-\alpha\lambda_i| \end{aligned}$$

with $m_k \to \infty$, then $|\phi_i^{(k)}| < |\lambda_i|$, so $\rho(H_k) < \rho(R)$ for $k = 0, 1, 2, \cdots$.

Let $\{y_k\}_{k=0}^{\infty}$ be generated by the iteration sequence $y_k = Ry_{k-1} + c$ from (2.1) with the same initial value x_0, and $x_0 - x^*$ be an eigenvector of R with eigenvalue λ_i for $|\lambda_i| = \rho(R)$, then

$$\begin{aligned}
\|x_{k+1} - x^*\| &= \|H_k H_{k-1} \cdots H_0 (x_0 - x^*)\| \\
&= \|\phi_i^{(k)} \phi_i^{(k-1)} \cdots \phi_i^{(0)} (x_0 - x^*)\| \\
&\leq \|\rho(H_k)\rho(H_{k-1}) \cdots \rho(H_0)(x_0 - x^*)\| \\
&< \|\rho(R)\rho(R) \cdots \rho(R)(x_0 - x^*)\| \\
&= \|R^{k+1}(x_0 - x^*)\| = \|y_{k+1} - x^*\|,
\end{aligned}$$

we complete the proof. □

Remark 3. From Theorem 3.1 we know that the conclusions are also held for an appropriate m_k. For example, let $\alpha = 0.3$ and $\rho(R) = 0.8$, then $|\alpha\lambda_i| \leq 0.24$, if $m_k = 16$, then $|\alpha\lambda_i|^{m_k - 1} = |\alpha\lambda_i|^{15} \leq 5.04 \times 10^{-10}$.

4. The accelerated inner-outer iteration method and precondition

4.1 The accelerated inner-outer iteration method

In this section, we combine the Algorithm 1 with other solvers as an acceleration scheme, and obtain the accelerated inner-outer iteration method. At first we can use the inner-outer iterations, then switch to other solvers when the inner iterations converge fast.

Algorithm 2: The accelerated inner-outer iteration method

Input: $R,\ c,\ \alpha,\ \varepsilon,\ m$

Output: x

1:$x \leftarrow c$

2: $z \leftarrow Rx$

3: **while** $\|c - x + z\|_2 \geq \varepsilon$

4: $\quad g \leftarrow (1 - \alpha)z + c$

5: $\quad$ **for** $i = 1 : \kappa$

6: $\quad\quad x \leftarrow g + \alpha z$

7: $\quad\quad z \leftarrow Rx$

8: $\quad$ **end**

9: $\quad$ **If** $i \leq \kappa,\ x = z + c,\ z = Rx$; **return**

10: **end while**

11. $x \leftarrow z + c$

In Algorithm 2 we use a for loop to count the number of the inner iterations. In line 9, we carry out the iteration sequences derived from the splitting $A = M - N$ with the latest x as an initial value.

4.2 Preconditioning for nonstationary iteration methods

Let $A = M-N$ be a convergence splitting, then we can use $(I-\alpha R)^{-1}M^{-1}$ as a preconditioner for the linear system (1.1). Since the computation of the preconditioner is as difficult to solve (1.1), then we use the Neumann series approximation of $(I-\alpha R)^{-1}M^{-1}$ as a preconditioner. If $0 < \alpha < 1$ and $\rho(R) < 1$, then from Lemmas 3.1 and 3.2 we have

$$(I-\alpha R)^{-1} = \sum_{i=0}^{\infty}(\alpha R)^i. \tag{4.1}$$

We can obtain an approximation of $(I-\alpha R)^{-1}M^{-1}$ by adopting m terms from (4.1):

$$(I-\alpha R)^{-1}M^{-1} \approx (I+\alpha R+(\alpha R)^2+\cdots+(\alpha R)^m)M^{-1}. \tag{4.2}$$

Let λ_i $(i = 1,\cdots n)$ be an eigenvalue of R and $A = M - N$. If we precondition (1.1) with (4.2), then we have

$$\begin{aligned} Q &= \left(I+\alpha R+(\alpha R)^2+\cdots+(\alpha R)^m\right)M^{-1}A \\ &= \left(I+\alpha R+(\alpha R)^2+\cdots+(\alpha R)^m\right)M^{-1}(M-N) \\ &= \left(I+\alpha R+(\alpha R)^2+\cdots+(\alpha R)^m\right)(I-R) \end{aligned}$$

and

$$\tilde{\lambda}_i = (1-\lambda_i)(1+\alpha\lambda_i+\cdots+(\alpha\lambda_i)^m), \tag{4.3}$$

where $\tilde{\lambda}_i$ $(i = 1,\cdots n)$ is an eigenvalue of the matrix Q.

In order to show the effectiveness of the preconditioner (4.2) on the krylov subspace methods, now we will analyze the distribution of the eigenvalues of Q.

From (4.3) we have

$$\begin{aligned} \left|\tilde{\lambda}_i - 1\right| &= \left|\tfrac{(1-\lambda_i)(1-(\alpha\lambda_i)^{m+1})}{1-\alpha\lambda_i} - 1\right| \\ &= \left|\tfrac{(\alpha\lambda_i)^{m+1}(\lambda_i-1)+\lambda_i(\alpha-1)}{1-\alpha\lambda_i}\right| \\ &< \tfrac{2(\alpha\rho(R))^{m+1}+\rho(R)(1-\alpha)}{1-\alpha\rho(R)}, \end{aligned} \tag{4.4}$$

then all the eigenvalues of Q are scattered in a circle centered at (1,0) with the radius $\frac{2(\alpha\rho(R))^{m+1}+\rho(R)(1-\alpha)}{1-\alpha\rho(R)}$. Let $\alpha = \rho(R)$, then from (4.4) we get

$$\left|\tilde{\lambda}_i - 1\right| < \frac{2\rho(R)^{2m+2}+\rho(R)(1-\rho(R))}{1-\rho(R)^2}.$$

If $\rho(R) = 0.7$ and $m = 2$, then $\frac{2\rho(R)^{2m+2}+\rho(R)(1-\rho(R))}{1-\rho(R)^2} = 0.8731$.

Remark 4. Let $A = \tilde{M}_i - \tilde{N}_i$ $(i = 1, 2)$ be two convergence splittings and $\rho(\tilde{M}_1^{-1}\tilde{N}_1) < \rho(\tilde{M}_2^{-1}\tilde{N}_2)$, then from (4.4) we obtain

$$\frac{2(\alpha\rho(\tilde{R}_1))^{m+1}+\rho(\tilde{R}_1)(1-\alpha)}{1-\alpha\rho(\tilde{R}_1)} < \frac{2(\alpha\rho(\tilde{R}_2))^{m+1}+\rho(\tilde{R}_2)(1-\alpha)}{1-\alpha\rho(\tilde{R}_2)},$$

so the preconditioned matrix Q based on $(\tilde{M}_1, \tilde{N}_1)$ may be more effective.

5. The inner-outer iteration method for the matrix equation $AXB = C$

Firstly we will review some well-known definitions and lemmas, which can be found in [1,6].

Let X be an $m \times n$ matrix, then $\mathrm{vec}(X)$ is defined to be a column vector of size $m \cdot n$ made of the columns of X stacked atop one another from left to right. Let A and B be $m \times n$ and $p \times q$ matrices, respectively. Then the Kronecker product $A \otimes B$ is the $(m \cdot p) \times (n \cdot q)$ matrix

$$\begin{bmatrix} a_{11} \cdot B & \cdots & a_{1n} \cdot B \\ \vdots & & \vdots \\ a_{m1} \cdot B & \cdots & a_{mn} \cdot B \end{bmatrix}.$$

Lemma 5.1. Let A, B and X be $m \times m$, $n \times n$, and $m \times n$ matrices, respectively. Then the following properties hold:

1. $\mathrm{vec}(AX) = (I_n \otimes A) \cdot \mathrm{vec}(X)$;
2. $\mathrm{vec}(XB) = (B^T \otimes I_m) \cdot \mathrm{vec}(X)$.

Lemma 5.2. The following relations about Kronecker products hold:

1. Assume that the products $A \cdot C$ and $B \cdot D$ are well defined. Then $(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D)$;
2. if A and B are invertible, then $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$;
3. $(A \otimes B)^T = A^T \otimes B^T$.

Lemma 5.3. Let $\lambda(A)$ and $\mu(B)$ be the spectrums of $A \in R^{n \times n}$ and $B \in R^{m \times m}$, respectively, then

$$\lambda(A \otimes B) = \{\lambda_i \mu_j : \lambda_i \in \lambda(A), \mu_j \in \mu(B), i = 1, 2, \cdots, n; j = 1, 2, \cdots, m\}.$$

In [12] the authors proposed a class of iteration methods for solving the matrix equation

$$AXB = C. \tag{5.1}$$

Since the matrix equation (5.1) is very important in matrix theory and applications, hence how to effectively solve this kind of equation has been under research recently, many iteration methods [9,13,14,16] have been proposed for solving (5.1) and corresponding matrix equations.

Based on the matrix splittings of the matrix A ro B, we can obtain the corresponding inner-outer iteration method for solving (5.1). In this section we only consider the inner-outer iteration method for the matrix splittings of B.

Let $B^T = \hat{M} - \hat{N}$ be a convergent matrix splitting and A be nonsingular, then we can get the equivalent form of (5.1) as follows:

$$X(I - \hat{R}) = \hat{C}, \tag{5.2}$$

where $\hat{R} = \hat{N}^T(\hat{M}^T)^{-1},\ \hat{C} = A^{-1}C(\hat{M}^T)^{-1}$.

The outer iteration for (5.2):

$$X_{k+1}(I - \beta\hat{R}) = (1 - \beta)X_k\hat{R} + \hat{C},\ k = 0, 1, 2, \cdots \tag{5.3}$$

with $0 < \beta < 1$.

The inner iteration for (5.3):

$$Y_{t+1} = \beta Y_t \hat{R} + \hat{G},\ t = 0, 1, 2, \cdots, p-1 \tag{5.4}$$

with $\hat{G} = (1-\beta)X_k\hat{R} + \hat{C}$. Here we take $Y_0 = X_k$ as the initial value and assign Y_p as the new X_{k+1}.

Similar to Algorithm 1, the inner-outer iteration method for (5.1) can be described as follows:

Algorithm 3: The inner-outer iteration method for (5.1)

Input: $\hat{R},\ \hat{C},\ \beta,\ \delta,\ \zeta$

Output: X

1:$X \leftarrow \hat{C}$

2: $Y \leftarrow X\hat{R}$

3: **while** $\|\hat{C} - X + Y\|_2 \geq \delta$

4: $\quad \hat{G} \leftarrow (1-\beta)Y + \hat{C}$

5: $\quad$ **repeat**

6: $\quad\quad X \leftarrow \hat{G} + \beta Y$

7: $\quad\quad Y \leftarrow X\hat{R}$

8: $\quad$ **until** $\|\hat{G} + \beta Y - X\|_2 < \zeta$

9: **end while**

10: $X \leftarrow Y + \hat{C}$

Now we will analyze the convergence of the inner-outer iterations (5.3) and (5.4). By Lemma 5.1 we firstly transform the matrix equation (5.1) into the following form:

$$(B^T \otimes A)x = \bar{c}, \tag{5.5}$$

where x and $\bar{c}$ are the forms of vec(X) and vec(C). From the matrix splitting $B^T = \hat{M} - \hat{N}$ and Lemma 5.2 , we have

$$(I - \bar{R})x = \hat{c}, \tag{5.6}$$

where $\bar{R} = \hat{M}^{-1}\hat{N} \otimes I$ and $\hat{c} = (\hat{M} \otimes A)^{-1}\bar{c}$.

The inner-outer iterations for (5.6):

$$(I - \beta\bar{R})x_{k+1} = (1-\beta)\bar{R}x_k + \hat{c},\ k = 0, 1, 2, \cdots \tag{5.7}$$

$$y_{s+1} = \beta\bar{R}y_s + \hat{g},\ s = 0, 1, 2, \cdots, p-1 \tag{5.8}$$

with $\hat{g} = (1-\beta)\bar{R}x_k + \hat{c}$.

From Theorems 2.3, 2.4 and Lemma 5.3, we can get the following convergent theorems for (5.7) and (5.8) as well as (5.3) and (5.4), respectively.

Theorem 5.1. Given $\rho(\hat{M}^{-1}\hat{N}) < 1$ and $0 < \beta < 1$, then the outer iteration (5.7) is convergent.

Proof. Since $\rho(\hat{M}^{-1}\hat{N}) < 1$, then from Lemma 5.3 we have $\rho(\bar{R}) = \rho(\hat{M}^{-1}\hat{N} \otimes I) < 1$. The proof is completed similar to that of Theorem 2.3. □

Theorem 5.2. If $\rho(\hat{M}^{-1}\hat{N}) < 1$ and $0 < \beta < 1$, then the inner iteration (5.8) is convergent.

Proof. From Lemma 5.3 and the assumptions we obtain $\rho(\bar{R}) = \rho(\hat{M}^{-1}\hat{N} \otimes I) < 1$, then we complete the proof according to Theorem 2.4. □

Remark 5. If we rewrite the inner-outer iterations (5.7)-(5.8) as a two-stage iteration framework just like (3.1), we can also get the corresponding conclusions similar to Theorem 3.1.

6. Numerical results

In this section, we perform some numerical examples to illustrate the effectiveness of our algorithms in this paper. The numerical experiments are performed in Matlab R2010 on an Intel dual core processor (2.30 GHz, 4GB RAM). We use four iteration parameters to test these iteration methods, which are iteration step (denoted as IT), computing time in seconds (denoted as CPU), number of matrix-vectors (denoted as MV), and relative residual (denoted as RES) defined by $\frac{\|r_k\|_2}{\|b\|_2}$ with $r_k = b - Ax_k$. Three sparse matrices P that we will use in the following examples are listed in Table 1, where "Average Nonzeros" means the average number of the nonzero elements per row. For each matrix we will deal with it by some techniques, then each matrix P is a column-stochastic matrix. Let $A = I - \varphi P$, where I is an identity matrix and $0 < \varphi < 1$, then A is a nonsingular $M-$matrix.

Table 1: Three test matrices for (1.1).

Name	Size	Nonzeros	Average Nonzeros
Minnesota	2,642× 2,642	6,606	2.50
Wb-cs-stanford	9,914×9,914	36,854	3.71
Stanford-Berkeley	683,446×683,446	7,583,376	11.0

Example 1. In this example, we use the AOR method, the inner-outer (IO) method (3.1) and the accelerated inner-outer (AIO) method (Algorithm 2) based on the AOR splitting (2.5) for solving (1.1), respectively, where $m_k = 2$ and $\kappa = 2$. The test matrix P is Minnesota matrix (available from http://www.cise.ufl.edu/research/sparse/matrices/Gleich/index.html) and $\varphi = 0.95$, $b(i) = 1$ $(i = 1, \cdots, n)$. All algorithms are terminated once the residual norms are below 10^{-8}.

Numerical results are reported in Tables 2, 3. From Table 2 we find that AIO method perform better than the AOR and IO methods for different ω, γ in both iteration number and CPU time, but it needs more MV compared with other two methods except $\omega = 1.8$ and $\gamma = 1.7$. Moreover, all the iteration methods have more efficiency when ω, γ are near 1.5. From Table 3 we also get the similar conclusions for the SOR, IO and AIO methods, where the IO and AIO methods are derived from the matrix splitting (2.5) with $\omega = \gamma$.

Example 2. This example is devoted to comparing the iteration number, CPU time and MV of the IO method for different η and α, and the IO method is based on the Richardson iteration. The test matrix P is the Stanford-Berkeley matrix of order 683,446 (available from http://cise.ufl.edu/research/sparse/matrices/SNAP/email-Enron.html) and $\varphi = 0.95$, $b = (1 -$

Table 2: Numerical results of the AOR, IO and AIO methods

ω	γ	$\rho(R)$	Iteration method	IT	MV	CPU	RES
			The AOR method	114	114	4.7011	8.79×10^{-9}
1.8	1.7	0.8446	The IO method	32	64	1.5123	8.38×10^{-9}
			The AIO method	23	69	1.2242	4.44×10^{-9}
			The AOR method	56	56	2.4827	7.29×10^{-9}
1.6	1.5	0.7010	The IO method	30	60	1.5081	7.56×10^{-9}
			The AIO method	23	69	1.2694	7.14×10^{-9}
			The AOR method	70	70	3.0767	8.59×10^{-9}
1.5	1.4	0.7506	The IO method	40	80	1.8571	6.40×10^{-9}
			The AIO method	30	90	1.5610	7.64×10^{-9}
			The AOR method	136	136	5.5520	9.19×10^{-9}
1.2	1.1	0.9398	The IO method	76	152	3.5596	9.26×10^{-9}
			The AIO method	57	171	2.9287	8.60×10^{-9}
			The AOR method	244	244	10.256	9.61×10^{-9}
0.9	0.8	0.9268	The IO method	136	272	6.2308	9.64×10^{-9}
			The AIO method	101	303	5.1983	9.67×10^{-9}

Table 3: Numerical results of the SOR, IO and AIO methods

ω	γ	$\rho(R)$	Iteration method	IT	MV	CPU	RES
			The SOR method	67	67	2.8872	8.22×10^{-9}
1.7	1.7	0.7457	The IO method	32	64	1.5238	6.55×10^{-9}
			The AIO method	20	60	1.0836	9.46×10^{-9}
			The SOR method	57	57	2.5109	8.66×10^{-9}
1.5	1.5	0.6814	The IO method	32	64	1.5286	9.58×10^{-9}
			The AIO method	24	72	1.3163	5.98×10^{-9}
			The SOR method	121	121	5.2262	9.02×10^{-9}
1.2	1.2	0.8545	The IO method	68	136	3.1752	8.30×10^{-9}
			The AIO method	51	153	2.7113	7.96×10^{-9}
			The SOR method	224	224	9.8298	9.80×10^{-9}
0.9	0.9	0.9204	The IO method	125	250	5.8433	9.67×10^{-9}
			The AIO method	90	270	4.8832	9.74×10^{-9}
			The SOR method	276	276	11.818	9.49×10^{-9}
0.8	0.8	0.9350	The IO method	154	308	7.0680	9.26×10^{-9}
			The AIO method	114	342	5.8866	9.72×10^{-9}

$\varphi)v/n$, where $v(i) = 1$ $(i = 1, \cdots, n)$. All iterations are terminated once the residual norms are below 10^{-8}.

From the discussion of η in [3,21], a very strict η may take a long time to compute the inner iteration in Algorithm 1, and just to implement a single outer iteration once a time. Setting η very loose, however, may make the inner iteration do not sufficiently approximate the exact solution of (2.4), and this may slow the convergence rate overall.

From Table 4 we find that the IO method may converge faster for a smaller η, but take more computing time and MV for a given stopping criterion, for example $\eta = 10^{-6}$. On the contrary, for a looser η the IO method has a slower convergence than other four values of η, such as $\eta = 10^{-2}$ or $\eta = 10^{-3}$.

In table 5 we perform the IO method with different α and $\eta = 10^{-4}$. From Table 5 we observe that the IO method is not sensitive to the choice of α in CPU time. When α is close to 1, it needs less iteration number, but has more MV at the same time.

Table 4: Numerical results of the IO method for the Stanford-Berkeley matrix with different η.

$\alpha = 0.5$	Iter	MV	CPU	RES
$\eta = 10^{-2}$	422	422	20.382	9.11×10^{-9}
$\eta = 10^{-3}$	422	422	20.338	9.22×10^{-9}
$\eta = 10^{-4}$	410	440	20.685	9.34×10^{-9}
$\eta = 10^{-5}$	389	516	22.353	9.33×10^{-9}
$\eta = 10^{-6}$	367	663	25.668	9.14×10^{-9}

Table 5: Numerical results of the IO method for the Stanford-Berkeley matrix with different α.

$\eta = 10^{-4}$	Iter	MV	CPU	RES
$\alpha = 0.2$	420	431	20.636	9.04×10^{-9}
$\alpha = 0.4$	414	438	20.711	9.11×10^{-9}
$\alpha = 0.5$	410	440	20.740	9.34×10^{-9}
$\alpha = 0.7$	401	445	20.489	9.13×10^{-9}
$\alpha = 0.9$	389	445	20.506	9.14×10^{-9}

Example 3: This example mainly illustrates the effectiveness of the preconditioner (4.2) for

the linear system (1.1) by using the GMRES method, where we choose $m = 2$ and $\varphi = 0.99$. The test matrices P are the Minnesota matrix and Wb-cs-stanford matrix which has the order of 9,914 (available from http://www.cise.ufl.edu/research/sparse/matrices/Gleich/index.html.), and $b(i) = 1$ $(i = 1, \cdots, n)$. In this example we use three preconditioners, which are the preconditioner (4.2) based on the Jacobi splitting and the Gauss-Seidel (GS) splitting, and the ILU preconditioner with no fill-in (ILU(0)), respectively.

Fig.1 shows the distribution of the eigenvalues of the matrix Q for the preconditioners based on the Jacobi and GS splittings for different α, respectively. we find that the matrix Q for the preconditioner(GS) has more eigenvalues concentrated around 1, which is more obvious for the larger α, so the PGMRES method with the preconditioner(GS) is more efficient than the PGMRES method with the preconditioner(Jacobi), which is consistent with Remark 4. From Table 6 we learn that the PGMRES method with the three preconditioners all outperforms the GMRES method in both IT and CPU time, and the PGMRES(ILU(0)) performs best. From Table 7 we notice that the PGMRES(GS) has the same iteration number as the PGMRES(ILU(0)) for $\alpha = 0.8$, while the PGMRES(ILU(0)) takes more CPU time than the PGMRES(GS) and PGMRES(Jacobi) for difficult α, respectively.

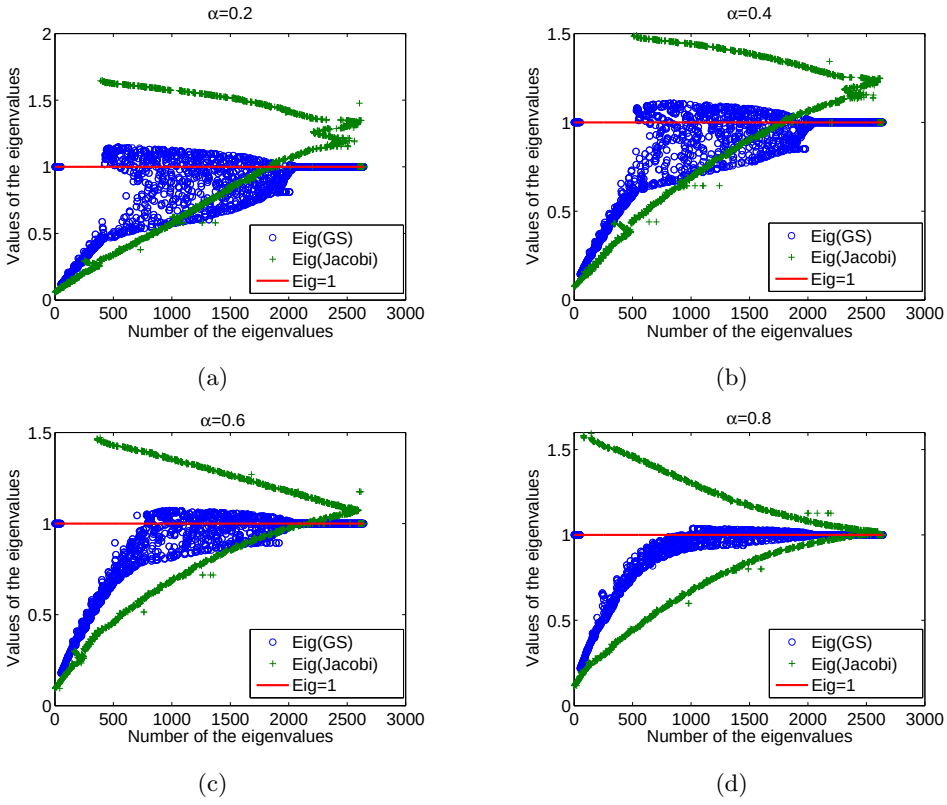

Figure 1: The distribution of eigenvalues for the minnesota matrix

Example 4. In this example A and B are strictly diagonally dominant matrix, $A(i,i) = 6$, $A(i,i+1) = 3$, $A(i+1,i) = -1$, $A(i,i+2) = -1$ and $B(i,i) = 8$, $B(i,i+1) = -1$, $B(i+1,i) =$

Table 6: Numerical results for the minnesota matrix

α	m	Iteration method	IT	CPU	RES
0.2	2	PGMRES(Jacobi)	108	1.3747	8.55×10^{-10}
		PGMRES(GS)	89	1.0794	9.83×10^{-10}
0.4	2	PGMRES(Jacobi)	91	1.1233	9.35×10^{-10}
		PGMRES(GS)	74	0.8820	8.89×10^{-10}
0.6	2	PGMRES(Jacobi)	81	0.9832	9.09×10^{-10}
		PGMRES(GS)	62	0.7342	7.91×10^{-10}
0.8	2	PGMRES(Jacobi)	76	0.9078	8.95×10^{-10}
		PGMRES(GS)	53	0.6140	7.17×10^{-10}
		PGMRES(ILU(0))	42	0.4689	9.53×10^{-10}
		GMRES	131	1.8129	8.76×10^{-10}

Table 7: Numerical results for the Wb-cs-stanford matrix

α	m	Iteration method	IT	CPU	RES
0.2	2	PGMRES(Jacobi)	121	1.7350	9.20×10^{-10}
		PGMRES(GS)	79	1.5441	8.76×10^{-10}
0.4	2	PGMRES(Jacobi)	96	1.1428	8.99×10^{-10}
		PGMRES(GS)	64	1.0750	9.08×10^{-10}
0.6	2	PGMRES(Jacobi)	79	0.8412	9.62×10^{-10}
		PGMRES(GS)	53	0.7468	7.06×10^{-10}
0.8	2	PGMRES(Jacobi)	71	0.6610	9.36×10^{-10}
		PGMRES(GS)	45	0.6469	6.34×10^{-10}
		PGMRES(ILU(0))	45	1.7448	7.76×10^{-10}
		GMRES	159	2.0648	9.63×10^{-10}

$-1,\ B(i,i+2) = -1$ for $i = 1, 2, \cdots, n$; $C_{ij} = 1$ for $i, j = 1, 2, \cdots, n$.

We compare the IO method based on the Jacobi splitting with the Jacobi-type method and the HSS method, respectively, where we apply the IO method similar to (3.1) with $\beta = 0.5$ and $m_k = 2$. From Table 8 we find that the IO method is more efficient than the Jacobi-style method and HSS method in term of iteration number and CPU time, respectively.

Table 8: Numerical results for $AXB = C$

	Jacobi-type method			HSS method			The IO method ($m_k = 2$)		
n	IT	CPU	RES	IT	CPU	RES	IT	CPU	RES
200	97	0.1017	5.42×10^{-10}	62	0.6162	6.38×10^{-10}	29	0.0511	5.74×10^{-10}
500	100	1.6448	5.71×10^{-10}	65	9.5407	9.24×10^{-10}	30	1.0152	4.30×10^{-10}
800	100	6.1598	2.63×10^{-10}	67	16.021	7.08×10^{-10}	30	3.7903	5.44×10^{-10}
1000	101	11.277	2.94×10^{-10}	67	36.703	4.20×10^{-10}	30	6.5638	2.39×10^{-10}

Example 5. In this example A is an $M-$matrix defined by

$$A = \begin{bmatrix} 10 & -1 & -2 & -1 & -3 & -1 \\ -3 & 11 & -2 & -1 & -3 & -2 \\ -1 & -2 & 15 & -5 & -2 & -3 \\ -3 & -4 & -1 & 14 & -2 & -2 \\ -3 & -5 & -1 & -3 & 16 & -1 \\ -1 & -2 & -3 & -4 & -2 & 16 \end{bmatrix},$$

and $b = [1\ 1\ 1\ 1\ 1\ 1]^T$.

We apply the preconditioned IO method for solving (1.1), which is based on the Jacobi and Gauss-Seidel splittings, respectively. We use the preconditioner (1.4) [23], and let $\alpha = 0.8,\ m_k = 2$. From Tables 9,10 we find that the preconditioned IO method outperforms other three iteration methods in term of IT and CPU time. Moreover, the preconditioned IO method (GS) has more effectiveness than the preconditioned IO method (Jacobi), and performs best among all the iteration methods in this example.

7. Conclusion

In this paper, we give an inner-outer iteration method for solving the linear system (1.1). Furthermore, the inner-outer iterations can also be used as a preconditioner for the Krylov

Table 9: Numerical results of Example 5

Iteration method	IT	CPU	RES
Jacobi method	136	0.000819	8.77×10^{-10}
Preconditioned Jacobi method	113	0.000682	8.25×10^{-10}
The IO method (Jacobi)	82	0.000563	9.88×10^{-10}
The preconditioned IO method (Jacobi)	68	0.000336	9.04×10^{-10}

Table 10: Numerical results of Example 5

Iteration method	IT	CPU	RES
Gauss-Seidel method	68	0.000337	7.45×10^{-10}
Preconditioned Gauss-Seidel method	59	0.000247	7.92×10^{-10}
The IO method (GS)	39	0.000225	8.12×10^{-10}
The preconditioned IO method (GS)	32	0.000173	7.06×10^{-10}

subspace method such as GMRES method. We also apply the inner-outer iteration method for solving the matrix equation $AXB = C$, which is more efficient than the iteration methods [12]. Future work may further improve the IO iteration method and construct more effective algorithms for solving (1.1) and (5.1), respectively.

Acknowledgements

The work is supported by the China Scholarship Council (201706935029), the Key Laboratory of Surveying and Mapping Technology on Island and Reef, National Administration of Surveying, Mapping and Geoinfomation (2013A03) and National key scientific instrument and equipment development projects (2013YQ120343).

References

[1] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[2] Z.Z. Bai, Sharp error bounds of some Krylov subspace methods for non-Hermitian linear systems, Appl. Math. Comput. 109 (2000) 273-285.

[3] D.F. Gleich, A.P. Gray, C. Greif, T.Lau, An inner-outer iteration method for computing PageRank, SIAM J. Sci. Comput. 32(2010) 349C371.

[4] Z.Z. Bai, M. Benzi, F. Chen, Modified HSS iteration methods for a class of complex symmetric linear systems, Computing. 87 (2010) 93-111.

[5] P. Concusand, G.H. Golub, A generalized conjugate gradient method for non-symmetric systems of linear equations, in Computing Methods in Applied Sciences and Engineering, Lecture Notes in Econom. and Math. Systems 134, R. Glowinski and J.R. Lions, eds, Springer-Verlag, Berlin, (1976) 56-65.

[6] J.W. Demmel, Applied numerical linear algebra.Philadelphia: Society for Industrial and Applied Mathematics, 1997.

[7] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.

[8] Z.Z. Bai, G.H. Golub, M.K. Ng, Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, SIAM J. Matrix Anal. Appl. 24 (2003) 603-626.

[9] F.X. Zhang, Y. Li, W.B. Guo, J.L. Zhao, Least squares solutions with special structure to the linear matrix equation AXB=C, Appl. Math. Comput. 217 (2011) 10049-10057.

[10] A. Hadjidimos, Successive overrelaxation (SOR) and related methods, J. Comput. Appl. Math. 123 (2000) 177-191.

[11] Z.Z. Bai, G.H. Golub, J.Y. Pan, Preconditioned Hermitian and skew-Hermitian splitting methods for non-Hermitian positive semidefinite linear systems, Numer. Math. 98 (2004) 1C32.

[12] Z.L. Tian, M.Y. Tian, Z.Y. Liu, T.Y. Xu, The Jacobi and Gauss-Seidel-type iteration methods for the matrix equation $AXB = C$, Appl. Math. Comput. 292 (2017) 63-75.

[13] Z.Y. Peng, An iterative method for the least squares symmetric solution of the linear matrix equation AXB=C, Appl. Math. Comput. 170 (2005) 711-723.

[14] H.M. Zhang, A finite iterative algorithm for solving the complex generalized coupled Sylvester matrix equations by using the linear operators, J. Franklin. Inst. 354 (2017) 1856-1874.

[15] Z.Z. Bai, G.H. Golub, L.Z. Lu, Block triangular and skew-Hermitian splitting methods for positive-definite linear systems, SIAM J. Sci. Comput. 26 (2005) 844C863.

[16] H.M. Zhang, H.C. Yin, Conjugate gradient least squares algorithm for solving the generalized coupled Sylvester matrix equations, Comput. Math. Appl. 73 (2017) 2529-2547.

[17] B.Y. Pu, T.Z. Huang, C. Wen, A preconditioned and extrapolation-accelerated GMRES method for PageRank, Appl. Math. Lett. 37 (2014) 95-100.

[18] G.H. Golub, C. Grief, Arnoldi-type algorithms for computing stationary distribution vectors with application to PageRank, BIT. 46 (2006) 759-771.

[19] D.J. Evans, M.M. Martins, M.E. Trigo, The AOR iterative method for new preconditioned linear systems, J. Comput. Appl. Math. 132 (2001) 461-466.

[20] Y. Zhang, T.Z. Huang, Modified iterative methods for nonnegative matrices and M-matrices linear systems, Comput. Math. Appl. 50 (2005) 1587-1602.

[21] C.Q. Gu, F. Xie, K. Zhang, A two-step matrix splitting iteration for computing PageRank, J. Comput. Appl. Math. 278 (2015) 19-28.

[22] J.P. Milaszewicz, Improving Jacobi and Gauss-Seidel iterations, Linear Algebra Appl. 93 (1987) 161-170.

[23] A. Hadjidimos, D. Noutsos, M. Tzoumas, More on modifications and improvements of classical iterative schemes for M-matrices, Linear Algebra Appl. 364 (2003) 253-279.

[24] A. Hadjimos, Accelerated overrelaxation method, Math.Comp. 32 (1978) 149-157.

[25] Y.Z. Song, On the convergence of the MAOR method, J. Comput. Appl. Math. 79 (1997) 299-317.